NAME

jar − Java archive tool

SYNOPSIS

jar [**−C**] [**c**] [**f**] [**i**] [**J**] [**M**] [**m**] [**O**] [**t**] [**u**] [**v**] [**x** *file*] [*manifest-file*]
destination input-file [*input-files*]

DESCRIPTION

The **jar** tool is a Java application that combines multiple files into a single JAR archive file. It is also a general-purpose archiving and compression tool, based on ZIP and the ZLIB compression format. However, **jar** was designed mainly to facilitate the packaging of Java applets or applications into a single archive. When the components of an applet or application (.class files, images and sounds) are combined into a single archive, they can be downloaded by a Java agent (like a browser) in a single HTTP transaction, rather than require a new connection for each piece. This dramatically improves download time. The **jar** tool also compresses files, which further improves download time. In addition, it allows individual entries in a file to be signed by the applet author so that their origins can be authenticated. The syntax for the **jar** tool is almost identical to the syntax for the **tar**(1) command. A **jar** archive can be used as a class path entry, whether or not it is compressed.

The three types of input files for the **jar** tool are:

- Manifest file (optional)
- Destination **jar** file
- Files to be archived

Typical usage is:

example% jar cf myjarfile *.class

In this example, all the class files in the current directory are placed in the file named **myjarfile**. A manifest file is automatically generated by the **jar** tool and is always the first entry in the **jar** file. By default, it is named **META-INF/MANIFEST.MF**. The manifest file is the place where any meta-information about the archive is stored. Refer to the *Manifest Format* in the **SEE ALSO** section for details about how meta-information is stored in the manifest file.

To use a pre-existing manifest file to create a new **jar** archive, specify the old manifest file with the **m** option:

example% jar cmf myManifestFile myJarFile *.class

When you specify **cfm** instead of **cmf** (that is, you invert the order of the **m** and **f** options), you need to specify the name of the jar archive first, followed by the name of the manifest file:

example% jar cfm myJarFile myManifestFile *.class

The manifest uses RFC822 ASCII format, so it is easy to view and process manifest-file contents.

OPTIONS

The following options are supported:

−C Changes directories during execution of the **jar** command. For example:

example% jar uf foo.jar −C classes *

c Creates a new or empty archive on the standard output.

f The second argument specifies a **jar** file to process. In the case of creation, this refers to the name of the **jar** file to be created (instead of on **stdout**). For table or xtract, the second argument identifies the **jar** file to be listed or extracted.

i Generates index information for the specified jar file and its dependent **jar** files. For example,

example% jar i foo.jar

would generate an **INDEX.LIST** file in **foo.jar** which contains location information for each package in

foo.jar and all the **jar** files specified in **foo.jar**'s Class-Path attribute.

J option Pass *option* to the Java virtual machine, where *option* is one of the options described on the man page for the java application launcher, java(1). For example, *−J-Xms48m* sets the startup memory to 48 megabytes. It is a common convention for **−J** to pass options to the underlying virtual machine.

M Does not create a manifest file for the entries.

m Includes manifest information from specified pre-existing manifest file. Example use:

example% jar cmf myManifestFile myJarFile *.class

You can add special-purpose name-value attribute headers to the manifest file that are not contained in the default manifest. Examples of such headers are those for vendor information, version information, package sealing, and headers to make JAR-bundled applications executable. See the *JAR Files* trail in the *Java Tutorial* and the *JRE Notes for Developers* web page for examples of using the **m** option.

O Stores only, without using ZIP compression.

t Lists the table of contents from standard output.

u Updates an existing JAR file by adding files or changing the manifest. For example:

example% jar uf foo.jar foo.class

adds the file **foo.class** to the existing JAR file **foo.jar**, and

example% jar umf foo.jar

updates **foo.jar**'s manifest with the information in manifest.

v Generates verbose output on **stderr**.

x *file* Extracts all files, or just the named files, from standard input. If *file* is omitted, then all files are extracted; otherwise, only the specified file or files are extracted.

If any of the *files* is a directory, then that directory is processed recursively.

EXAMPLES

To add all of the files in a particular directory to an archive:

```
example% ls
0.au      3.au      6.au      9.au          at_work.gif
1.au      4.au      7.au      Animator.class  monkey.jpg
e.au      5.au      8.au      Wave.class      spacemusic.au
example% jar cvf bundle.jar *
adding: 0.au
adding: 1.au
adding: 2.au
adding: 3.au
adding: 4.au
adding: 5.au
adding: 6.au
adding: 7.au
adding: 8.au
adding: 9.au
adding: Animator.class
adding: Wave.class
adding: at_work.gif
adding: monkey.jpg
adding: spacemusic.au
example%
```

If you already have subdirectories for images, audio files, and classes that already exist in an HTML directory, use **jar** to archive each directory to a single **jar** file:

```
example% ls
audio classes images
example% jar cvf bundle.jar audio classes images
adding: audio/1.au
adding: audio/2.au
adding: audio/3.au
adding: audio/spacemusic.au
adding: classes/Animator.class
adding: classes/Wave.class
adding: images/monkey.jpg
adding: images/at_work.gif
example% ls −l
total 142
drwxr-xr-x   2 brown    green       512 Aug  1 22:33 audio
-rw-r--r--   1 brown    green     68677 Aug  1 22:36 bundle.jar
drwxr-xr-x   2 brown    green       512 Aug  1 22:26 classes
drwxr-xr-x   2 brown    green       512 Aug  1 22:25 images
example%
```

To see the entry names in the **jar** file using the **jar** tool and the **t** option:

```
example% ls
audio bundle.jar classes images
example% jar tf bundle.jar
META-INF/MANIFEST.MF
audio/1.au
audio/2.au
audio/3.au
audio/spacemusic.au
classes/Animator.class
classes/Wave.class
images/monkey.jpg
images/at_work.gif
example%
```

To display more information about the files in the archive, such as their size and last modified date, use the **v** option:

```
example% jar tvf bundle.jar
145 Thu Aug 01 22:27:00 PDT 1996 META-INF/MANIFEST.MF
946 Thu Aug 01 22:24:22 PDT 1996 audio/1.au
1039 Thu Aug 01 22:24:22 PDT 1996 audio/2.au
993 Thu Aug 01 22:24:22 PDT 1996 audio/3.au
48072 Thu Aug 01 22:24:23 PDT 1996 audio/spacemusic.au
16711 Thu Aug 01 22:25:50 PDT 1996 classes/Animator.class
3368 Thu Aug 01 22:26:02 PDT 1996 classes/Wave.class
12809 Thu Aug 01 22:24:48 PDT 1996 images/monkey.jpg
527 Thu Aug 01 22:25:20 PDT 1996 images/at_work.gif
example%
```

If you bundled a stock trade application (applet) into the following **jar** files,

main.jar buy.jar sell.jar other.jar

and you specified the **Class-Path** attribute in **main.jar**'s manifest as

Class-Path: buy.jar sell.jar other.jar

then you can use the **i** option to speed up your application's class loading time:

example$ jar i main.jar

An **INDEX.LIST** file is inserted in the **META-INF** directory which will enable the application class loader to download the right **jar** files when it is searching for classes or resources.

SEE ALSO

keytool(1)

JAR Files @
http://java.sun.com/docs/books/tutorial/jar/

JRE Notes @
http://java.sun.com/j2se/1.4/runtime.html#example

JAR Guide @
http://java.sun.com/j2se/1.4/docs/guide/jar/index.html

For information on related topics, use the search link @
http://java.sun.com/